Chapter 18 Service-oriented software engineering:

18.1 Service-oriented architecture

18.2 RESTful services

18.3 Service engineering

18.4 Service composition

Note: Images excluded due to time constraints

18.1 Service-Oriented Architecture (SOA)

Overview

- **Service-Oriented Architecture (SOA)** is an architectural style that utilizes executable services in applications.
- Services have well-defined, published interfaces, and multiple providers may offer the same service.

Structure of SOA

- Service providers design, implement, and publish services in a registry.
- Service requestors (clients) discover services from the registry and bind their applications to them.
- Communication occurs through standard service protocols, supported by a stack of key international standards.

Key International Standards

1. **SOAP**: Standard for message interchange that defines essential and optional components of messages.
2. **WSDL (Web Service Description Language)**: Describes service interfaces, operations, parameters, and bindings.
3. **WS-BPEL**: Workflow language standard for defining process programs involving multiple services.

4. **UDDI (Universal Description, Discovery, and Integration)**: Meant for service discovery but has fallen out of use in favor of standard search engines.

Supporting Standards

1. **WS-Reliable Messaging**: Ensures messages will be delivered once and only once.
2. **WS-Security**: Specifies security policies and digital signatures.
3. **WS-Addressing**: Represents address information in SOAP messages.
4. **WS-Transactions**: Coordinates transactions across distributed services.

Message Exchange

- Services in SOA communicate by exchanging XML-based messages over standard Internet transport protocols like HTTP and TCP/IP.

WSDL Service Description

- Specifies what the service does (interface), how it communicates (binding), and where to find it (endpoint).

Limitations

- WSDL descriptions lack information about service semantics and non-functional characteristics.
- Manual understanding of service functionality and performance is often required.

Conclusion

- SOA leverages a range of international and supporting standards for effective service discovery, binding, and interaction.
- While it offers flexibility in service choice and binding, it also demands careful understanding and validation of service functionalities.

18.2 RESTful Services

Overview

- **RESTful Services** are a "lightweight" alternative to standard SOAP-based web services.
- They are suitable for single-function services with simple I/O interfaces and are commonly used in mobile devices.

REST Architecture

- **REST** stands for Representational State Transfer, focusing on transferring resource representations between server and client.
- Resources are central to RESTful architecture, identified uniquely by a URL.

Resource Operations

1. **Create**: Done using the POST action, brings a resource into existence.

2. **Read**: Done using the GET action, returns the resource's value.
3. **Update**: Done using the PUT action, modifies the resource's value.
4. **Delete**: Done using the DELETE action, removes the resource.

Data Access

- RESTful services use HTTP or HTTPS protocols with actions limited to POST, GET, PUT, and DELETE.
- Data can be accessed using URLs, sometimes with URL queries for specific information.

Data Representation

- RESTful services support multiple data formats like JSON and XML, providing more efficiency in data processing.

Stateless Design

- RESTful services should be stateless; all required state information should be returned to the requestor.

Advantages

- Lower overhead and better performance, especially on devices with limited processing capabilities.
- Easier to implement alongside existing websites.

Limitations

1. **Complex Interfaces**: Difficult to represent services with complex interfaces.
2. **Lack of Standard Description**: Requires reliance on informal documentation.
3. **Quality of Service**: Must build your own infrastructure for monitoring and managing quality and reliability.

SOAP vs REST

- It's possible to offer both SOAP-based and RESTful interfaces to the same service, allowing clients to choose the best-suited method.

Conclusion

- RESTful services are increasingly popular due to their lightweight nature and performance benefits but come with their own set of limitations and challenges.

18.3 Service Engineering

Overview

- Service engineering focuses on creating services for reuse in service-oriented applications.
- The process involves three logical stages: Service candidate identification, Service design, and Service implementation and deployment.
- Services can be started from existing components or legacy systems.

Three Logical Stages in Service Engineering

1. **Service Candidate Identification**: Identification of potential services and defining their requirements.
    - Involves understanding an organization's business processes.
    - Three types of services: Utility services, Business services, and Coordination or process services.
    - Questions are posed to ensure the service is logically coherent, independent, and reusable.
2. **Service Design**: Designing the logical and implementation interfaces for the service.
    - Options between SOAP-based and RESTful services.
    - Importance of defining exceptions and operation details.
3. **Service Implementation and Deployment**: Actual coding and testing of the service, making it available for use.
    - Can be done in programming languages like Java or C#.
    - Services need to be tested for conformity and functional behavior before deployment.

Types of Services

1. **Utility Services**: General functionality used by different business processes.
2. **Business Services**: Specific to a business function.
3. **Coordination or Process Services**: Support more general business processes.

Questions for Identifying Service Candidates

- Is the service associated with a single resource used in different business processes?
- Is the task carried out by different people in the organization?
- Is the service independent?
- Does the service have to maintain state?
- Might the service be used by external clients?
- Are different versions of the service needed for different nonfunctional requirements?

Legacy Systems

- Services can act as "wrappers" for legacy systems, enabling their integration with modern systems.

Service Deployment and Documentation

- Once deployed, documentation is essential for potential users.
- Documentation may include business information, functionality description, usage guide, and subscription information.

Challenges and Complexities

- Decision-making between RESTful and SOAP-based approaches can be complex.
- Natural language descriptions in service specifications are subject to misinterpretation. Ontology-based specifications are not widely used due to their complexity.

18.4 Service Composition

Overview

- **Service Composition** is the fundamental principle of service-oriented software engineering where services are composed and configured to create new, composite services.
- These composite services may be part of web applications or other service compositions.
- Companies are increasingly adopting this model for intra- and inter-organizational applications.
- The process involves considerations for service failures, user demands, and workflow steps.

Types of Services Involved

1. **Specially Developed Services**: Created specifically for the application.
2. **Business Services**: Developed within a company for various internal uses.
3. **External Services**: Services from external providers, often integrated for extended functionality.

Application of Service Composition

- **Example Case**: An airline creating a vacation package service by composing services from hotel booking agencies, car rental companies, and local attractions.
- **Workflow**: A sequence of steps or activities carried out in time order, often representing a business process.

Complexity and Challenges

- Handling service failures through **exception management**.
- Dealing with **non-standard user demands**, like special accessibility needs.
- **Compensation actions** to handle incompatibilities between service executions.

Key Stages in Service Composition

1. **Formulate Outline Workflow**: Create an abstract design based on requirements.
2. **Discover Services**: Search for existing services that can be included in the composition.
3. **Select Possible Services**: Choose services based on functionality, cost, and quality.
4. **Refine Workflow**: Detail the workflow based on selected services.
5. **Create Workflow Program**: Transform the abstract workflow into an executable program.
6. **Test Completed Service**: Test the composite service, more complicated when external services are involved.

Workflow Design and Implementation

- **Workflow Design Notations**: UML or BPMN, which can be automatically converted into executable models.
- **Iterative Design**: The model may undergo multiple iterations for optimal reuse of available services.

Testing Service Compositions

- **Challenges**:
    i. Lack of control over external services.
    ii. Dynamic binding causing inconsistency.

iii. Variable non-functional behavior due to other users.
iv. Cost implications based on payment models.
v. Difficulty in simulating service failures for testing.

- **Importance of Testing**: Vital for ensuring both functional and non-functional requirements are met. Resolving these testing challenges remains an ongoing research issue.

Summary

- Service-oriented architecture is an approach to software engineering where reusable, standardized services are the basic building blocks for application systems.
- Services may be implemented within a service-oriented architecture using a set of XML-based web service standards. These include standards for service communication, interface definition, and service enactment in workflows.
- Alternatively, a RESTful architecture may be used, which is based on resources and standard operations on these resources. A RESTful approach uses the http and https protocols for service communication and maps operations on the standard http verbs POST, GET, PUT, and DELETE.
- Services may be classified as utility services that provide a general-purpose functionality, business services that implement part of a business process, or coordination services that coordinate the execution of other services.
- The service engineering process involves identifying candidate services for implementation, defining the service interface, and implementing, testing, and deploying the service.
- The development of software using services is based on the idea that programs are created by composing and configuring services to create new composite services and systems.
- Graphical workflow languages, such as BPMN, may be used to describe a business process and the services used in that process. These languages can describe interactions between the organizations that are involved.